Software Design Document

for

Artificial Intelligence and Data Science for Air Pollution Prediction and Visualization (AIDSAPPV)

Version 1.2 approved

Prepared by Micky Chan, Eric Chan, Peter Gatsby, Larry Gutierrez, Jose Landa

NASA and LA City

December 7th, 2020

Revision History

Name	Date	Reason For Changes	Version
Team	12/7/2020	First draft of SDD	1.0.0

<Add rows as necessary when the document is revised. This document should be consistently updated and maintained throughout your project. If ANY requirements are changed, added, removed, etc., immediately revise your document.>

1. Introduction

1.1 Purpose

This document is to explain in detail the functions that the following applications will perform. The document will inform readers as to what the applications will do. The purpose of these applications is to provide users with general information about air quality and air pollution in their area. These applications will also provide predictions on the levels of air pollution and air quality using machine learning and data science techniques.

1.2 Document Conventions

Roman numerals will indicate which application is being talked about, the website or mobile application. Followed by bullet points to indicate what each section is talking about. External links will be underlined in blue

1.3 Intended Audience and Reading Suggestions (NEEDS UPDATE)

The intended audience of the software requirements specification document are developers and project managers. It is suggested to look at the table of contents first in order to find any topics that you may be looking for. If not, quickly skimming through the document will allow the reader to obtain a general understanding of the project. If you are a developer or project manager, it is suggested to read Section 2 to Section 4 to understand the design and the architecture of the project.

1.4 System Overview

I. **Air Pollution in Los Angeles County Data Visualization (Web App)**
   - This application is used to show the measured levels of air pollutants as well as some common causes of air pollution around Los Angeles County. Currently, we plan on using data and feature layers from ArcGIS's Living Atlas to display current conditions on air quality and air pollution. Moving forward, we plan on using data gathered from various APIs in conjunction with data science and machine learning techniques to generate live and accurate measurements as well as generating predictions.

II. **Air Pollution Personalized App (Android App) (NEEDS UPDATE)**
   - The application is used to show the measured levels of air pollutants around Los Angeles County. This is done by using feature layers that rely on sensors around Los Angeles County. Currently for the mobile app, we plan on using data from ArcGIS's Database to measure certain levels of different air pollutants. Moving forward we plan on integrating machine learning techniques into the software to be able to not only provide real time data, but forecasted air quality data as well.

2. Design Considerations

The following section details the Assumptions and Dependencies, General Constraints, Goals and Guidelines as well as the Development Method of the applications.

2.1 Assumptions and Dependencies

I. **Air Pollution in Los Angeles County Data Visualization (Web App)**
   - The following software uses the ArcGIS Online JavaScript API and assumes that any browser allows the proper permissions for it to work
   - The following software is reliant on the ArcGIS Living Atlas and assumes that the Feature Layers used remain public and in the same format
   - The following software is reliant on using ArcGIS Living Atlas to gather data and assumes developers have an ArcGIS license
   - The following software is developed by gathering data from various APIs and assumes that API keys are provided to the developers

II. **Air Pollution Personalized App (Android App) (NEEDS UPDATE)**
   - The following application assumes the user will have a compatible phone/android version
   - The following software assumed the data remain updating, to display on the app
   - The following software is developed using ArcGIS and Esri and assumes that all developers have a ArcGIS license

2.2 General Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

Listed below are the general constraints for each application of this project.

I. **Air Pollution in Los Angeles County Data Visualization (Web App)**
   - **Software Environment**
     - Developers are required to have a functioning ArcGIS license
     - Developers are required to have basic knowledge of HTML
     - Developers are required to have basic knowledge of CSS

        - Developers are required to have basic knowledge of the JavaScript language
        - Developers are required to have knowledge of the React framework
    - **End-User Environment**
        - A web browser is required to use and view the web application
        - A mouse, keyboard, monitor, and desktop/laptop are required for user inputs and outputs
    - **Interoperability requirements**
        - Data is gathered from ArcGIS's Living Atlas
        - Data is gathered from various APIs
    - **Performance requirements**
        - Application should load maps and their datasets in less than 5 seconds
        - Application should load graphs in less than 5 seconds
        - Application should load news articles in less than 5 seconds

II. **Air Pollution Personalized App (Android App) (NEEDS UPDATE)**

- **Software Environment**
    - Developers are required to have a functioning ArcGIS license
- **End-User Environment**
    - A mobile device is required to use and view the application
- **Interoperability requirements**
    - Data is gathered from ArcGIS's Layer Database
    - Data is gathered from various sensors surrounding Los Angeles County
- **Performance requirements**
    - Application should load maps and their datasets in less than 5 seconds

2.3 Goals and Guidelines

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own subsubsection if you wish. Such goals might be:

I. **Air Pollution in Los Angeles County Data Visualization**

- The data gathered from ArcGIS's Living Atlas should be accurate
- The data gathered from the various APIs should be accurate
- The GUI should be user-friendly
- The application should be able to effectively visualize air pollution data for users

II. Air Pollution Personalized App (NEEDS UPDATE)

- The data retrieved from the ArcGIS database should be accurate
- The GUI should be user-friendly
- The application should be able to effectively visualize air pollution data for users

2.4 Development Methods

The method to develop the two applications in AIDSAPPV is similar to the Agile Development Method. The developers have split into two teams; one team is responsible for the development of the web application while the other team is responsible for the mobile application. Both teams split their tasks and requirements among their members to effectively complete the multitude of components that the project requires. Regular, weekly meetings are conducted to assess the progress of both applications and decide what components or modules should be finished next.

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

These would be things such as the ‘Water Fall Development’ methods, ‘Agile Development’, ‘Unplanned Mad Scramble Development’, or other development models and variations. Describe how these were applied in the case of your project.

3. Architectural Strategies

Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

I. Air Pollution in Los Angeles County Data Visualization (Web App)

- Use of a particular type of product (programming language, database, library, etc.)
    - ArcGIS
    - JavaScript
    - HTML
    - CSS
    - React
    - Python
    - pandas library
    - scikit-learn library
- Reuse of existing software components to implement various parts/features of the system
    - This software is the first version
- Future plans for extending or enhancing the software
    - Adding predictions of air quality conditions
    - Collect data in real time to implement into the application
- User interface paradigms
    - A computer and internet connection is required to use the application as well as a mouse to navigate the web application

II. Air Pollution Personalized App (Android App) (NEEDS UPDATE)

-

- Use of a particular type of product (programming language, database, library, etc. ...)
- Reuse of existing software components to implement various parts/features of the system
- Future plans for extending or enhancing the software
- User interface paradigms (or system input and output models)
- Hardware and/or software interface paradigms
- Error detection and recovery
- Memory management policies
- External databases and/or data storage management and persistence
- Distributed data or control over a network
- Generalized approaches to control
- Concurrency and synchronization
- Communication mechanisms
- Management of other resources

Each significant strategy employed should probably be discussed in its own subsection. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose).

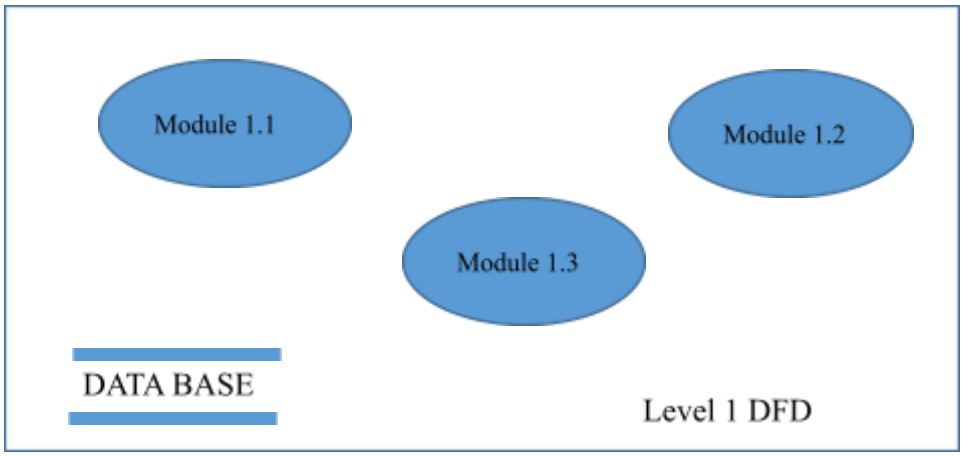
4. System Architecture

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Don't go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

This is where the level 0 DFD will probably work best.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its modules/components/subsystems (identifying each top-level modules/component/subsystem and the roles/responsibilities assigned to it).

Each subsection (i.e. “4.1.3 The ABC Module”) of this section will refer to or contain a detailed description of a system software component.



Level 1 Data Flow Diagrams (DFD) and Control Flow Diagrams (CFD) should probably go here.

Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them. Diagrams that describe a particular component or subsystem in detail should be included within the particular subsection that describes that component or subsystem.

5. Policies and Tactics

This section details the choice of specific products that were used, plans for ensuring requirements traceability, as well as plans for testing the software.

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures), but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Such decisions might concern (but are not limited to) things like the following (Must include 5.1, 5.2, and 5.3. The rest of these categories or custom ones can be added as needed.):

5.1 Choice of which specific products used

- Programming languages used:
    - Python
    - Java
    - JavaScript
- ArcGIS
- Android Studio

5.2 Plans for ensuring requirements traceability

The requirements of the project shall be traceable via the source code posted using GitHub version control. Requirements will be checked frequently in coordination with the software requirements specification document.

…Describe…

5.3 Plans for testing the software

The software shall be tested as requirements for the software are completed and as features and functionalities are added.

5.4 Plans for maintaining the software

The plans to maintain the software include collecting more data as well as updating data in real-time

…Describe…

5.# Engineering trade-offs

…Describe…

5.# Coding guidelines and conventions

…Describe…

5.# The protocol of one or more subsystems, modules, or subroutines

…Describe…

5.# The choice of a particular algorithm or programming idiom (or design pattern) to implement portions of the system's functionality

…Describe…

5.# Plans for maintaining the software

…Describe…

5.# Interfaces for end-users, software, hardware, and communications

…Describe…

5.# Hierarchical organization of the source code into its physical components (files and directories).

…Describe…

5.# How to build and/or generate the system's deliverables (how to compile, link, load, etc.)

…Describe…

5.# Describe tactics such as abstracting out a generic DatabaseInterface class, so that changing the database from MySQL to Oracle or PostGreSQL is simply a matter of rewriting the DatabaseInterface class.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own "best" judgement to decide this. There will usually be some global policies and tactics that should be discussed here, but decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.

6. Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Each subsection of this section will refer to or contain a detailed description of a system software component. The discussion provided should cover the following software component attributes:

This is where Level 2 (or lower) DFD's will go. If there are any additional detailed component diagrams, models, user flow diagrams or flowcharts they may be included here.

6.1 Data Visualization

6.1.1 Responsibilities

The primary responsibility of this module is to visualize data on a map and translate the data into UI elements such as symbols or polygons. This module should also ensure that all symbols are unique and that all polygons are colored based on intensity.

The primary responsibilities and/or behavior of this component. What does this component accomplish? What roles does it play? What kinds of services does it provide to its clients? For some components, this may need to refer back to the requirements specification.

6.1.2 Constraints

Some datasets used for this module are not up to date or do not update in real-time which may lead to inaccurate representation. Datasets that are up to date may also have null fields which may cause certain data to not appear on the map.

Any relevant assumptions, limitations, or constraints for this component. This should include constraints on timing, storage, or component state, and might include rules for interacting with this component (encompassing preconditions, post conditions, invariants, other constraints on input or output values and local or global values, data formats and data access, synchronization, exceptions, etc.)

6.1.3 Composition

The list below details the use and meaning of the subcomponents used in this module

- **Legend**: shows the different symbols and their meanings as well as the different colors for the polygons and their meanings
- **Toggle on/off:** allows for datasets to be toggled on/off and refreshes the map
- **Add dataset:** allows for custom datasets to be added to the map given they are in the right format
- **Switch map:** allows for multiple maps with different datasets and different usage to be shown in the same application

A description of the use and meaning of the subcomponents that are a part of this component.

6.1.4 Uses/Interactions

This module will be used to visualize the common causes of air pollution, the measured levels of air pollutants, the current demographics of the populus, as well as statistics on the effects of air pollution surrounding Los Angeles county.

A description of this components collaborations with other components. What other components is this entity used by? What other components does this entity use (this would include any side-effects this entity might have on other parts of the system)? This concerns the method of interaction as well as the interaction itself. Object-oriented designs should include a description of any known or anticipated subclasses, superclass's, and metaclasses.

6.1.5 Resources

The resources this module uses are datasets of Feature Layers that are present within ArcGIS's Living Atlas as well as data gathered from various sensors surrounding Los Angeles county.

A description of any and all resources that are managed, affected, or needed by this entity. Resources are entities external to the design such as memory, processors, printers, databases, or a software library. This should include a discussion of any possible race conditions and/or deadlock situations, and how they might be resolved.

6.1.6 Interface/Exports

The visualized data will be displayed on the Air Pollution in Los Angeles County Data Visualization application via the use of the ArcGIS Online JavaScript API.

The set of services (classes, resources, data, types, constants, subroutines, and exceptions) that are provided by this component. The precise definition or declaration of each such element should be present, along with comments or annotations describing the meanings of values, parameters, etc. For each service element described, include (or provide a reference) in its discussion a description of its important software component attributes (Classification, Definition, Responsibilities, Constraints, Composition, Uses, Resources, Processing, and Interface).

Much of the information that appears in this section is not necessarily expected to be kept separate from the source code. In fact, much of the information can be gleaned from the source itself (especially if it is adequately commented). This section should not copy or reproduce information that can be easily obtained from reading the source code (this would be an unwanted and unnecessary duplication of effort and would be very difficult to keep up-to-date). It is recommended that most of this information be contained in the source (with appropriate

comments for each component, subsystem, module, and subroutine). Hence, it is expected that this section will largely consist of references to or excerpts of annotated diagrams and source code.

7. Detailed Lower level Component Design

Other lower-level Classes, components, subcomponents, and assorted support files are to be described here. You should cover the reason that each class exists (i.e. its role in its package; for complex cases, refer to a detailed component view.) Use numbered subsections below (i.e. "7.1.3 The ABC Package".) Note that there isn't necessarily a one-to-one correspondence between packages and components.

7.1 Name of Class or File

7.x.1 Classification

The kind of component, such as a subsystem, class, package, function, file, etc.

7.x.2 Processing Narrative (PSPEC)

A process specification (PSPEC) can be used to specify the processing details

7.x.3 Interface Description

7.x.4 Processing Detail

7.x.4.1 Design Class Hierarchy

Class inheritance: parent or child classes.

7.x.4.2 Restrictions/Limitations

7.x.4.3 Performance Issues

7.x.4.4 Design Constraints

7.x.4.5 Processing Detail For Each Operation

8. Database Design

Include details about any databases used by the software. Include tables and descriptions.

9. User Interface

The user interface is the application, from the point of view of the users. Do your classes and their interactions (the logical and process views) impose restrictions on the user interface? Would removing some of these restrictions improve the user interface? Use some form of user interface flow model to provide an overview of the UI steps and flows. Don't go into too much refinement. You should include screen shots or wireframe layouts of significant pages or dialog elements. Make sure to indicate which of the system level modules or components that each of these user interface elements is interacting with.

9.1 Overview of User Interface

Describe the functionality of the system from the user's perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback Information that will be displayed for the user. This is an overview of the UI and its use. The user manual will contain extensive detail about the actual use of the software.

9.2 Screen Frameworks or Images

These can be mockups or actual screenshots of the various UI screens and popups.

9.3 User Interface Flow Model

A discussion of screen objects and actions associated with those objects. This should include a flow diagram of the navigation between different pages.

10. Requirements Validation and Verification

Create a table that lists each of the requirements that were specified in the SRS document for this software.
For each entry in the table list which of the Component Modules and if appropriate which UI elements and/or low level components satisfies that requirement.
For each entry describe the method for testing that the requirement has been met.

11. Glossary

An ordered list of defined terms and concepts used throughout the document. Provide definitions for any relevant terms, acronyms, and abbreviations that are necessary to understand the SDD document. This information may be listed here or in a completely separate document. If the information is not directly listed in this section provide a note that specifies where the information can be found.

12. References

<List any other documents or Web addresses to which this SDD refers. These may include other SDD or SRS documents, user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

Brad Appleton <brad@bradapp.net> http://www.bradapp.net

https://www.cs.purdue.edu/homes/cs307/ExampleDocs/DesignTemplate_Fall08.doc